7. Variabelen in Python

Een variabele is een gereserveerd stuk geheugen. Een variabele heeft een waarde en een naam. De grootte van het stuk geheugen hangt af van het type. In Python worden variabelen niet gedeclareerd[1], het type wordt bepaald door de toekenning van een waarde. Voorbeelden:

```
aantal = 5         # gehele toewijzing
straal = 2.0       # komma getal
pi = 3.14          # komma getal
naam = “cirkel”    # string
```

In Python kan je meerdere variabelen tegelijk een waarde geven

A = B = C = 3   # A, B en C krijgen waarde 3

A, B, C = 1, 2.3, “Jan”   # A krijgt waarde 1, B waarde 2.3 en C “Jan”

Python kent 6 standaard data types:

- Getal
- String
- List
- Tuple
- Dictionary
- Logisch

Een logische variabele heeft waarde **True** of **False**

Python getallen

Er zijn drie getaltypes in Python:

- int: geheel getal
- float: komma getal
- complex: complex getal

Gehele getallen zijn long-integers. Zij worden bewaard in 4 bytes en kunnen waarden aannemen tussen -32,768 en 32,767. Je mag gehele getallen noteren als

[1] In ander talen zoals C bestaat er declaratie-instructies zoals *int A;*
Dit reserveert geheugen voor een variabele met naam A en type int.

A = 234 # normale schrijfwijze
B = 0x2F # hexadecimale schrijfwijze van 47
C = 0b001100 # binaire schrijfwijze van 12

Komma getallen schrijf je als

X = 0.234
Y = 2.34E-2 # = 2,34 x 10^{-2} = 0.0234

Complexe getallen zijn getallen van de vorm x+y.i, x en y zijn reëel en i^2=-1

Z = 2.3 + 4j
wiskundigen gebruiken symbool i, Python symbool j

Rekenkundige bewerkingen

Python kent volgende rekenkundige operatoren:

+	Optellen	7+2 = 9
-	Aftrekken	7-2 = -5
*	vermenigvuldigen	7*2 = 14
/	Deling	7/2 = 3.5
%	Rest van de deling	7%2 = 1
**	Macht	7**2 = 49
//	Deling van gehele getallen, naar beneden afgerond	7//2 = 3 -4//3 = -2

Sommige beperkte implementaties van MicroPython kennen alleen gehele getallen. Je kunt alleen een deling van gehele getallen gebruiken, niet de gewone. (// kan, / niet)

Verkorte schrijfwijze van enkele bewerkingen

De uitdrukking

a += 1

betekent: tel 1 op bij a. Dat is equivalent met de uitdrukking

a = a + 1

We kunnen hetzelfde doen voor bewerkingen *, -, / en %. De voorbeelden hieronder zijn uitgewerkt met Thonny, interactief.

```
>>> a = b = c = d = e = 10
>>> a += 1
>>> b *= 2
>>> c -= 3
>>> d /= 3
```

```
>>> e %= 3
>>> print (a,b,c,d,e)
11 20 7 3.333333 1
```

Eerst geven we a, b, c, d en e de waarde 10. We tellen 1 op bij a, vermenigvuldigen b met 2, trekken 3 af van c , delen d door 3 en e wordt de rest van e met 3. De resultaten worden onderaan afgedrukt.

Vergelijkingsoperatoren

Vergelijkingsoperatoren vergelijken twee getallen of variabelen. Het resultaat is een logische waarde, True of False

==	Gelijk aan?	(2==3) = False
!=	Niet gelijk aan	
>	Groter dan	
<	Kleiner dan	
>=	Groter dan of gelijk aan	
<=	Kleiner dan of gelijk aan	

Logische operatoren

Logische operatoren zijn berekeningen met logische uitdrukkingen en geven resultaat True of False.

A and B	True als A én B True zijn
A or B	True als A en/of B True is
Not A	True als A False is

Waarheidstabellen zijn schematische voorstellingen van de werking van deze operatoren:

A	B	A and B	A or B	not A
1	1	1	1	0
0	1	0	1	1
1	0	0	1	0
0	0	0	0	1

De voorbeelden hieronder zijn uitgewerkt met Thonny in interactieve modus

```
>>> A =(2==3)
>>> print (A)
 False
>>> print ((2<3)or(4<3))
 True
>>> print ((2<3)and(4<3))
 False
>>> print (not(2!=3))
 False
```

Figuur 29: logische operatoren

Gegevens invoer en uitvoer

Controllers worden meestal gebruikt in toepassingen zonder uitgebreid toetsenbord en zonder computerscherm. Tijdens het ontwikkelen van software kan het handig zijn om resultaten op het scherm te zetten of in te voeren met het toetsenbord. We gebruiken hiervoor opdracht print voor uitvoer of input voor invoer.

Print

Met **print** zetten we gegevens op het scherm. Dat kan een string zijn, een getal, een uitdrukking of een combinatie van strings en/of getallen.

```
>>> print ("dit is een string")
 dit is een string
>>> print (4+5)
 9
>>> print("het resultaat is: ", 2**3)
 het resultaat is:  8
>>> print ("som = %u, quotiënt = %f" %(2+3, 2/3))
 som = 5, quotiënt = 0.666667
```

Figuur 30: instructie print

In het laatste voorbeeld drukken we een string met plaatshouders af. %u verwijst naar een geheel getal, %f naar een kommagetal. De af te drukken getallen staan achteraan. Mogelijke plaatshouders zijn:

%c	Character
%s	string
%u	unsigned integer
%d	signed integer
%f	float
%x of %X	Hexadecimaal getal
%e of %E	Exponent notation

Je kunt ook het minimaal aantal plaatsen aangeven voor een getal.

- **%4u**: drukt af als een geheel getal en voorzie 4 plaatsen minimaal. Lege plaatsen worden blanco.
- **%04u**: drukt af als een geheel getal en voorzie 4 plaatsen minimaal. Lege plaatsen worden 0.
- **%06.2f**: drukt af als een kommagetal met twee cijfers na de komma. Voorzie 6 plaatsen minimaal. Lege plaatsen vooraan worden 0.

```
MicroPython v1.14 on 2021-02-05; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>>
>>> print ("%4u" %2)
   2
>>> print ("%04u" %2)
0002
>>> print ("%06.2u" %2)
000002
>>> print ("%06.2f" %2.1)
002.10
>>>
```

Figuur 31: print met opmaak

In een printstatement kan je niet-afdrukbare tekens opnemen. Enkele daarvan zijn:

- \n: begin een nieuwe lijn
- \tab: naar volgende tab-plaats
- \a: een belletje

Input

Instructie **input** vraagt invoer via het toetsenbord. Wil je een getal als invoer, dan moet je de string converteren.

```
Shell
>>>
>>> naam = input ("geef uw naam:  ")
geef uw naam:  Dirk
>>> print (naam)
Dirk
>>> leeftijd = int(input ("hoe oud ben jij?  "))
hoe oud ben jij?  67
>>> print (leeftijd)
67
```

Figuur 32: instructie input

Voor leeftijd heeft de gebruiker *67* ingevoerd. Python beschouwt dit als een string. Functie int() maakt er een geheel getal van.

rekenen met bits

getalstelsels

In het dagelijkse leven werken we met 10-delige getallen, die vormen een systeem met 10 cijfers. Zo kan je getal 2347 schrijven als

$$2347 = 7 + 4.10 + 3.10^2 + 2.10^3$$

Computers werken binair en gebruiken 2 cijfers, 0 en 1. We noemen ze binaire getallen. Met prefix 0b geven we aan dat een binaire schrijfwijze volgt.

$$0b1001110 = 0 + 1.2 + 1.2^2 + 1.2^3 + 0.2^4 + 0.2^5 + 1.2^6 = 158$$

Werken met binaire getallen is lastig, we korten ze af door de bits in groepjes van 4 te plaatsen. Met 4 bits kan je 16 verschillende getallen voorstellen, die schrijven we als

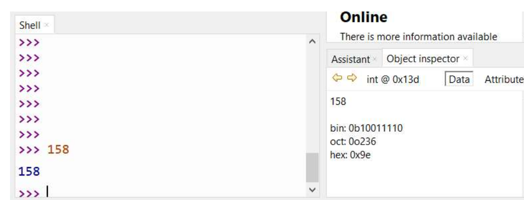
0000 = 0
0001 = 1
…
1001 = 9
1010 = A (=10)
…
1111 = F (=15)

Het binaire getal hierboven schrijven we als

0b1001110 = 0x9E = 14 + 9.16 = 158

Deze getallen noemen we hexadecimaal. We noteren ze met prefix 0x

Het omrekenen van talstelsels kan Thonny voor ons: geef een getal in het in teractieve venster (links onderaan). Rechts onderaan verschijnen de verschillende vormen.



Figuur 33: getalvormen met Thonny

In een programma kan je de verschillende schrijfwijzen gebruiken en met functie print() kan je afdrukken in de gewenste vorm.

Voor sommige toepassingen moeten we afzonderlijke bits van een binair getal manipuleren. We gebruiken hiervoor **bitwise operatoren.** Die werken met de overeenkomstige bits van e getallen.

- **bitwise AND**: c = a & b
  een bit van c is 1 als de overeenkomstige bits van a en b allebei 1 zijn.
  a = 0b10100111
  b = 0b11110000
  a & b = 0b10100000
- **bitwise OR**: c = a | b
  een bit van c is 1 als van het overeenkomstige bit van a en/of b 1 is.
  a = 0b10100111
  b = 0b11110000
  a | b = 0b11110111
- **bitwise NOT**: c = ~a
  een bit van c is 1 als het overeenkomstige bit van a 0 is,
  een bit van c is 0 als het overeenkomstige bit van a 1 is.
  a = 0b10100111
  ~a = 0b01011000
- **Bitshift right: >>** schuift alle bits van een byte/integer enkele plaatsen naar rechts. Lege plaatsen worden 0.
  a = 0b10100111
  a >> 3 = 0b00010100 // 3 plaatsen naar rechts
- **Bitshift left: <<** schuift alle bits van een byte/integer enkele plaatsen naar links. Lege plaatsen worden 0.
  a = 0b10100111
  a << 2 = 0b10011100 // 2 plaatsen naar links

Alle bits in een getal een plaats opschuiven naar links komt overeen met vermenigvuldigen met 2, opschuiven naar rechts betekent delen door 2.

Let op de volgorde van de bewerkingen:
- eerst * en / en %
- dan + en -
- dan vergelijken
- dan logische berekeningen
Bewerkingen met dezelfde prioriteit: van links naar rechts

1 + 2 * 3 // = 7 (eerst *, dan +)
1+2*3 == 7 && !(7 > 3) // = **false**